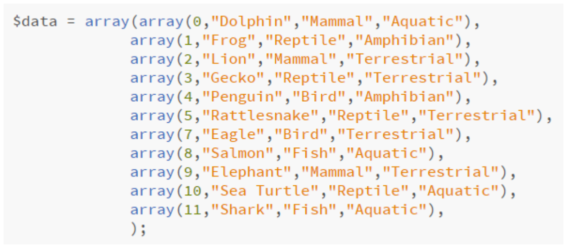
IDG2003 – Lab3

Exercise 1 : - Logical Operators, if/elseif/if statements, switch statements, for loops

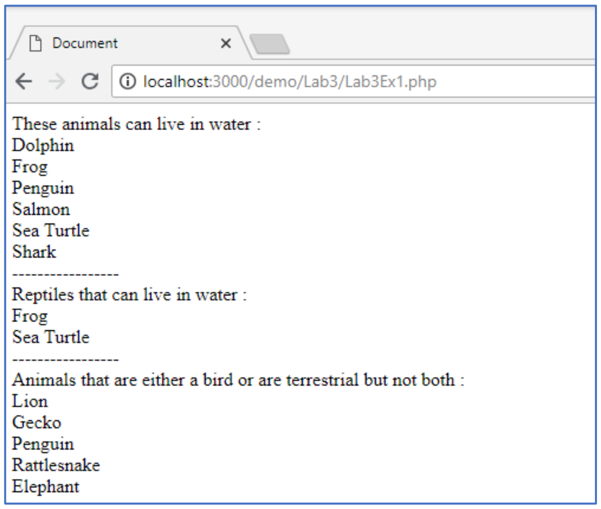
1. Given the array below, select the animals based on the following criteria:
   a. Animals that can live in water.
   b. Reptiles that can live in water.
   c. Animals that are either a bird or are terrestrial but not both.

   (Note, Amphibians can also live in water.)



```
$data = array(array(0,"Dolphin","Mammal","Aquatic"),
              array(1,"Frog","Reptile","Amphibian"),
              array(2,"Lion","Mammal","Terrestrial"),
              array(3,"Gecko","Reptile","Terrestrial"),
              array(4,"Penguin","Bird","Amphibian"),
              array(5,"Rattlesnake","Reptile","Terrestrial"),
              array(7,"Eagle","Bird","Terrestrial"),
              array(8,"Salmon","Fish","Aquatic"),
              array(9,"Elephant","Mammal","Terrestrial"),
              array(10,"Sea Turtle","Reptile","Aquatic"),
              array(11,"Shark","Fish","Aquatic"),
              );
```

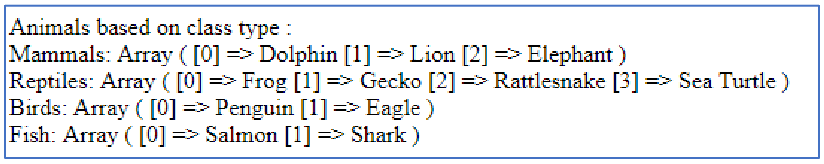
Expected Result :



```
These animals can live in water :
Dolphin
Frog
Penguin
Salmon
Sea Turtle
Shark
-----------------
Reptiles that can live in water :
Frog
Sea Turtle
-----------------
Animals that are either a bird or are terrestrial but not both :
Lion
Gecko
Penguin
Rattlesnake
Elephant
```

2. Create arrays for each of the class types : Mammals, Reptiles, Birds and Fish. Then use a for loop and if/elseif/else statements to populate the arrays with the correct animal names. Display the result of the populated arrays in the browser

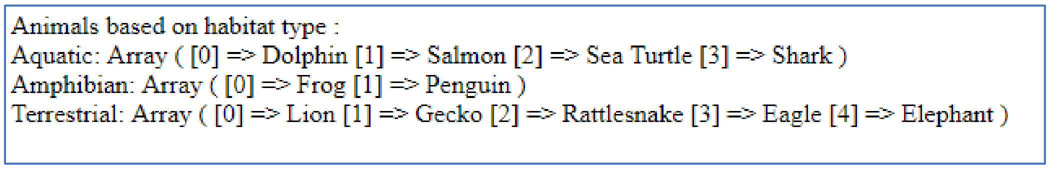
   Expected Result :



```
Animals based on class type :
Mammals: Array ( [0] => Dolphin [1] => Lion [2] => Elephant )
Reptiles: Array ( [0] => Frog [1] => Gecko [2] => Rattlesnake [3] => Sea Turtle )
Birds: Array ( [0] => Penguin [1] => Eagle )
Fish: Array ( [0] => Salmon [1] => Shark )
```

3. Similar to part 2, create arrays for each of the habitat types: Aquatic, Amphibian and Terrestrial. This time, populate the arrays using switch statements and display the result of the populated arrays in the browser.

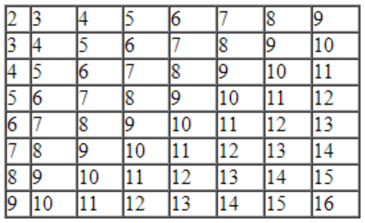
   Expected Result :



```
Animals based on habitat type :
Aquatic: Array ( [0] => Dolphin [1] => Salmon [2] => Sea Turtle [3] => Shark )
Amphibian: Array ( [0] => Frog [1] => Penguin )
Terrestrial: Array ( [0] => Lion [1] => Gecko [2] => Rattlesnake [3] => Eagle [4] => Elephant )
```

Exercise 2 : - Creating a checkboard – Nested for loops, ternary operator

1. Initialise the first 'for' loop to create the rows of an 8x8 table with the first row having an index of 1.
2. Nest a second 'for' loop inside the first to populate the columns in each row, with the first column having an index of 1.
3. Print the result for the sum of the row index and the column index in each cell of the table as shown below.

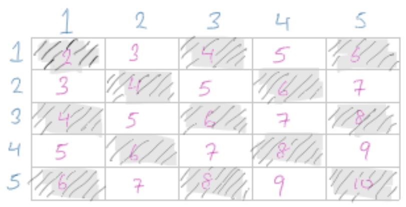


2	3	4	5	6	7	8	9
3	4	5	6	7	8	9	10
4	5	6	7	8	9	10	11
5	6	7	8	9	10	11	12
6	7	8	9	10	11	12	13
7	8	9	10	11	12	13	14
8	9	10	11	12	13	14	15
9	10	11	12	13	14	15	16

4. If instead of the printing the sum, you shaded the cells either black or white, can you create a checkboard pattern?
    a. You can use the statement below to shade the cell a specific color.

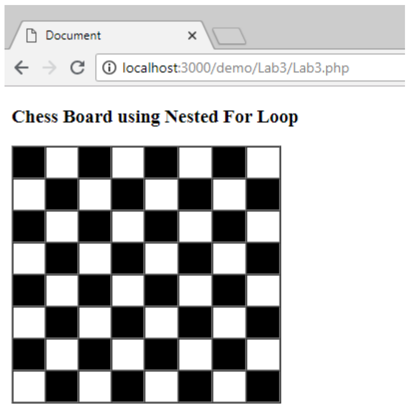
```php
echo "<td height=30px width=30px bgcolor=$color>"."</td>";
```

    b. Use a ternary operator to assign the variable $color, in the above statement, a value of either "#FFFFFF"(black) or "#000000"(white) based on some expression that would allow you to create a checkboard pattern. (hint: what pattern do you notice in the sum of (rowIndex and columnIndex) in the figure below?)



	1	2	3	4	5
1	▨ 2	3	▨ 4	5	▨ 6
2	3	▨ 4	5	▨ 6	7
3	▨ 4	5	▨ 6	7	▨ 8
4	5	▨ 6	7	▨ 8	9
5	▨ 6	7	▨ 8	9	▨ 10

Expected Result :



Document

localhost:3000/demo/Lab3/Lab3.php

Chess Board using Nested For Loop

Exercise 3 : - Bubble Sort Algorithm – While Loops, Break/Continue Statements

Using while loops(nested), implement the bubble sort algorithm to sort elements in an array in ascending order.

Example :

Array	x	y	Is y empty?	Is x < y ?	Action	
[$\boxed{5 , 8}$, 2 , 1 , 10]	5	8	No	TRUE	No Swapping, move filter	[5 , 8 , 2 , 1 , 10]
[5 , $\boxed{8 , 2}$, 1 , 10]	8	2	No	FALSE	Swap, move filter	[5 , 2 , 8 , 1 , 10]
[5 , 2 , $\boxed{8 , 1}$, 10]	8	1	No	FALSE	Swap, move filter	[5 , 2 , 1 , 8 , 10]
[5 , 2 , 1 , $\boxed{8 , 10}$]	8	10	No	TRUE	No Swapping, move filter	[5 , 2 , 1 , 8 , 10]
[5 , 2 , 1 , 8 , $\boxed{10\ ]}$	10		YES, break			

Essentially, the process involves checking the first 2 entries of an array. If the 2nd entry is the less than the first entry, the entries are swapped else do nothing.
The 2nd and 3rd entries are then compared in the next iteration, and so on.
This is better demonstrated with the moving filter in the above table.

The nested loop can be broken by checking if the end of array has been reached, meaning if y is empty in the moving filter. (hint : is_null($y) can be used here to here if $y is empty).

If the end of array has been reached -> sweep again from the beginning of the array, since the smallest value may not be in the right position yet.

To break from the main while loop, implement a condition that check if a swap occurred in the nested loop. This means, the outer loop continues until no more swaps occurs in the nested loop.

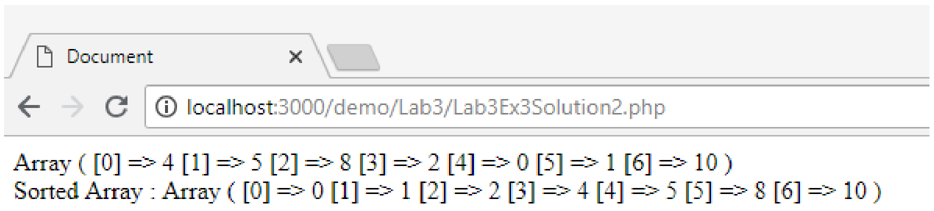
Note: this is a harder exercise and requires careful thought. I am not guiding you step by step this time, to allow you to figure things out by yourselves.

You can find more information about bubble sort here:

https://www.studytonight.com/data-structures/bubble-sort

Also note that in the link above, the algorithm given uses 'for' loops instead.
In this exercise however, you are expected to use 'while' loops and possibly also break/continue statements.

Result Expected : for an array [4,5,8,2,0,1,10] is as follows -



```
Array ( [0] => 4 [1] => 5 [2] => 8 [3] => 2 [4] => 0 [5] => 1 [6] => 10 )
Sorted Array : Array ( [0] => 0 [1] => 1 [2] => 2 [3] => 4 [4] => 5 [5] => 8 [6] => 10 )
```